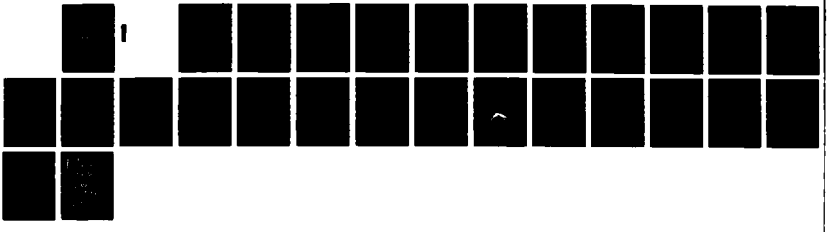
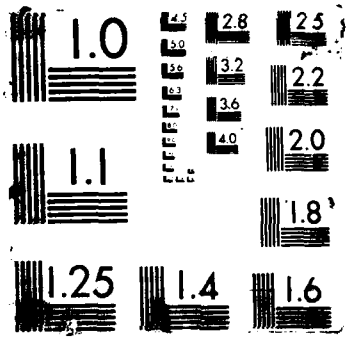
AD-A195 603 COMPLETE PIVOTAL GAUSSIAN ELIMINATION USING THE VETERANS' ADMINISTRATION FILE MANAGER(U) NAVAL HEALTH RESEARCH CENTER SAN DIEGO CA D R HODGINS 16 MAR 88 1/1

UNCLASSIFIED NHRC-88-12 F/G 12/3 NL




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3

COMPLETE PIVOTAL GAUSSIAN ELIMINATION USING THE VETERANS' ADMINISTRATION FILE MANAGER

Dallas R. Hodgins

Medical Information Systems Department
Naval Health Research Center
P. O. Box 85122
San Diego, CA 92138-9174

Report 88-12, supported by the Naval Medical Research and Development Command, Department of the Navy under work unit M0095.005-1053. The views expressed in this article are those of the author and do not reflect the official policy or position of the Department of the Navy, Department of Defense, nor The U.S. Government. Approved for public release; distribution unlimited.

The skills and patience of my colleague, Kathryn Medrano, were indispensable in the preparation of this manuscript.

Summary

The solution to sets of linear equations is fundamental to mathematical modeling. The choice of a programming language for the algorithm used can critically influence the efficiency of the methodology. The viability of the MUMPS programming language is demonstrated in developing an extremely conservative and accurate routine to solve linear systems using a complete pivotal Gaussian forward elimination strategy.

The technique of using a simple hashing scheme to avoid post-multiplicative matrices to identify variable domains permits tight, concise coding while guaranteeing maximum accuracy, given the conditions of the problem. Linear algebra notation combined with using VAFM in a relational context allows the analysis maximum freedom in data manipulation.

The Gauss algorithm presented in this paper solves any n by n system in n^3 steps. Additional solutions demand only n^2 operations. Complete pivotal Gaussian elimination is a "perfect" algorithm in a mathematical sense.

COMPLETE PIVOTAL GAUSSIAN ELIMINATION USING THE VETERANS' ADMINISTRATION FILE MANAGER

Dallas R. Hodgins
Naval Health Research Center

Introduction

The installation of the Naval Occupational Health Information Management System (NOHIMS) at Naval industrial facilities makes possible a valuable data resource for industrial hygienists, medical researchers, and epidemiologists. The Veterans' Administration File Manager (FM) component of NOHIMS manages this data linking the worker with his environment. Prior to implementation, a need to augment the statistical tools of FM was recognized. The first phase of this effort consisted of providing researchers with basic descriptive statistical algorithms.[1] These programs were written in MUMPS (Massachusetts General Hospital Utility Multi-Programming System) using linear algebra notation and embedded in FM (also written in MUMPS). A conscious choice was made to design routines to efficiently handle small data samples, realistically recognizing the magnitude of data handled and the response time requirements of shipyard industrial hygienists. Three fundamental issues, however, remained unresolved:

(1) Is MUMPS a reasonable choice for numerical analysis in statistical work?

(2) Is FM the right choice for statistical data structuring and manipulation?

(3) Assuming one uses MUMPS and FM, what is the best logical data structure?

These questions are addressed in this paper in a pragmatic fashion by developing an algorithm to solve linear equations. Of the two problems central to abstract linear algebra, solutions to $Ax=b$ and $Ax=\lambda x$, the first has a perfect algorithm (in the mathematical sense)--Gaussian elimination. Indeed, the basic algorithm of numerical linear algebra is Gaussian elimination with partial pivoting. This paper presents a complete pivotal strategy using a hashing technique to eliminate the need of postmultiplication by a permutation matrix to ascertain the original domains of the unknowns.

The overwhelming power, both memory and processing, of current machines has led to the use of "black box" algorithms that may or may not be suitable to the problem at hand. Numerical techniques designed to handle very large models cannot afford the luxury of the conservativeness of a complete pivotal strategy in the standard decomposition of the matrix A. In keeping with "Small is Beautiful", the numerical model presented is designed for problems encountered by medical researchers, industrial hygienists, and statisticians that are dealing with a handful of variables and at most a few hundred cases --not thousands. These people want to know if their mathematical model is stable and they want the best numerical approach to the solution. They are willing to sacrifice machine time for soundness and accuracy in their research. The use of linear algebra and the "view" of the data afforded by FM, used in a relational sense, coalesce to provide a powerful and practical environment for numerical statistical analysis.

Materials and Methods

This work was done using Intersystems MUMPS M/VX 3.1 on a Digital Equipment Corporation VAX 11/785 with FM version 17.07.

The mathematical formulation most commonly used in statistics to express relationships between two or more variables is the linear equation of the form

$$Y' = X_1 + X_2V_1 + X_3V_2 \ldots X_kV_k \qquad (1)$$

where Y' is to be predicted, $V_1, V_2, \ldots V_k$ are the known values from the observed data on which the predictions are to be based, and $X_1, X_2 \ldots X_k$ are the numerical coefficients to be determined. We find these by minimizing the sum of the squares $(Y-Y')^2$ where the Y's are the observed values and the Y primes are the calculated values. This is done by taking the partial derivatives of the expression for $(Y-Y')^2$ with respect to as many unknown coefficients X_1, X_2, etc. as we have in the relationship.

The "normal" equations for two independent variables are:

$$\begin{aligned} \Sigma NX_1 &+ (\Sigma V_1)X_2 + (\Sigma V_2)X_3 = \Sigma Y \\ (\Sigma V_1)X_1 &+ (\Sigma V_1^2)X_2 + (\Sigma V_1V_2)X_3 = \Sigma V_1Y \\ (\Sigma V_2)X_1 &+ (\Sigma V_1V_2)X_2 + (\Sigma V_2^2)X_3 = \Sigma V_2Y \end{aligned} \qquad (2)$$

where N is the number of observations, ΣV_1, ΣV_2 are the sums of the independent variables, and ΣV_1V_2 is the sum of the product of the independent variables and ΣY is the sum of the inhomogeneous terms.

More concisely, we can represent this system by Ax=b where A is the square matrix of the independent variable weights of the overdetermined system in (1), X is a column vector representing the unknown values we are seeking, and b is the column vector of inhomogeneous terms. Ax=b has a solution if and only if b lies in the vector space spanned by the column vectors of A (which we made orthogonal to the error vector (y-y') by setting the partial derivations to zero, hence the term "normal"). The solution $x=A^{-1}b$ will be demonstrated using a complete pivotal forward Gaussian elimination strategy. The matrix A is factored into lower and upper triangular matrices L and U such that LU=A, allowing us to solve for $A^{-1}b$, not A^{-1} itself. The original columns of the unknowns are ascertained at completion of the computation by a simple hashing technique based on their original position in the matrix. This avoids complicated bookkeeping needed to effect the post-multiplication by permutation matrices--the price of which, in the traditional algebraic solution, is precisely why one seldom, if ever, finds an algorithm that incorporates complete pivoting. We can afford the conservatism of complete pivoting, however, as we are expressly manipulating small systems.

Purely hypothetically, suppose it was desired to predict the effect on the average price of a mobile surgical tent of the number of operating tables and light fixtures to illuminate the tables. Using FM we would search (using the option SEARCH FILE ENTRIES) the file containing fiscal data for information relating to surgical tents. This data would be placed in the global ^DIBT for us by FM. Using the TRANSFER ENTRIES option we transfer the following data to the file SURGTENT (created by MODIFY FILE ATTRIBUTES prior to the transfer) from ^DIBT:

Table 1. Relation SURGTENT:

	Number of Tables (V_1)	Number of Lights (V_2)	Price (Y), Dollars
3-Tuple 1	3	2	33800
.	2	1	29300
.	4	3	38800
.	2	1	29200
	3	2	34700
	2	2	29900
	5	3	43400
3-Tuple 8	4	2	37900

Table 1 in our notation is X1(J,K) where X1 is the relation SURGTENT, J is the record number (the primary key--tuple 1, tuple 2, etc.) and K is the domain: K=1=Number of Tables, K=2=Number of Lights, K=3=Price. Therefore, X1(7,3) represents the table (relation) surgical tent, 3-tuple 7 (set), domain 3 (price)--more precisely--43,400 dollars.

Carefully multiplying and summing, we obtain the last line in Table 2:

Table 2. Independent Variables

V_1	V_2	Y	V_1Y	V_2Y	V_1^2	V_1V_2	V_2^2
3	2	33,800	101,400	67,600	9	6	4
2	1	29,300	58,600	29,300	4	2	1
4	3	38,000	155,200	116,400	16	12	9
2	1	29,200	58,400	29,200	4	2	1
3	2	34,700	104,100	69,400	9	6	4
2	2	29,900	59,800	59,800	4	4	4
5	3	43,400	217,000	130,200	25	15	9
4	2	37,900	151,600	75,800	16	8	4
ΣV_1=25	ΣV_2=16	ΣY=277,000	ΣV_1Y=906,100	ΣV_2Y=577,700	ΣV_1^2=87	ΣV_1V_2=55	ΣV_2^2=36

Inserting the sums into our "normal" equations we arrive at

$$\begin{aligned} 8X_1 + 25X_2 + 16X_3 &= 277000 \\ 25X_1 + 87X_2 + 55X_3 &= 906100 \\ 16X_1 + 55X_2 + 36X_3 &= 577700 \end{aligned} \qquad (3)$$

the "normal" equations for the overdetermined system X1(J,K). For reasons of exposition, a file NORMEQ (file 4003) was created to hold these values. Part of the data dictionary for NORMEQ looks like this:

CURRENT ENTRIES

```
NAME: EQ1    X1: 8     X2: 25    X3: 16    X4: 277000
NAME: EQ2    X1: 25    X2: 87    X3: 55    X4: 906100        (4)
NAME: EQ3    X1: 16    X2: 55    X3: 36    X4: 577700
```

which does not require too much imagination to visualize as the equations in (3).

Let us digress a moment to familiarize ourselves with the terms and concepts of Gaussian elimination to motivate the subsequent discussion of the algorithm for the multiple regression solution. Consider this simple system of three equations in three unknowns (from Strang[2]):

$$\begin{aligned} X_1 + X_2 + 2X_3 &= 1 \\ X_1 + 0 + 4X_3 &= -2 \\ 2X_1 + X_2 - 2X_3 &= 7 \end{aligned} \qquad (5)$$

which can be expressed as Ax=b where

$$A = \begin{bmatrix} 1 & 1 & 2 \\ 1 & 0 & 4 \\ 2 & 1 & -2 \end{bmatrix}, \quad x = \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix}, \text{ and } b = \begin{bmatrix} 1 \\ -2 \\ 7 \end{bmatrix}$$

Simple Elimination

Displaying our problem in the following manner

$$\begin{bmatrix} 1 & 1 & 2 \\ 1 & 0 & 4 \\ 2 & 1 & -2 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} = \begin{bmatrix} 1 \\ -2 \\ 7 \end{bmatrix}$$

we proceed as follows:

Step 1: (a) Divide the first element of the first row (the first pivot) into the first element of the second row; multiply the first row by this quotient (equals 1) and subtract the first row from the second; (b) divide the first element of the first row into the first element of the third row; multiply the first row by this quotient (equals 2) and subtract the result from the third row obtaining:

$$\begin{bmatrix} 1 & 1 & 2 \\ 0 & -1 & 2 \\ 0 & -1 & -6 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} = \begin{bmatrix} 1 \\ -3 \\ 5 \end{bmatrix}$$

Step 2: Divide the second element of the third row by the second element of the second row (the second pivot); multiply the second row by this quotient (equals 1) and subtract the second row from the third, obtaining:

$$\begin{bmatrix} 1 & 1 & 2 \\ 0 & -1 & 2 \\ 0 & 0 & -8 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} = \begin{bmatrix} 1 \\ -3 \\ 8 \end{bmatrix}$$

This completes the forward elimination allowing us to solve for X_3 immediately ($-8X_3 = 8$, $X_3 = -1$).

Step 3: Substituting the value of X_3 back into row two we obtain

$$-X_2 + 2(-1) = -3 \text{ or } X_2 = 1$$

Back substituting the values of X_3 and X_2 into row one we have

$$X_1 + 1 + 2(-1) = 1 \text{ or } X_1 = 2.$$

Elimination with Partial Pivoting

Solving the same system with a partial pivoting strategy we simply start the forward elimination by choosing as the pivot row that row which has the largest element (in an absolute sense) in the pivot column and repeat the process for choosing the pivots at each stage of the elimination.

$$A = \begin{bmatrix} 1 & 1 & 2 \\ 1 & 0 & 4 \\ 2 & 1 & -2 \end{bmatrix} \rightarrow \begin{bmatrix} 2 & 1 & -2 \\ 1 & 1 & 2 \\ 1 & 0 & 4 \end{bmatrix} \rightarrow \begin{bmatrix} 2 & 1 & -2 \\ 0 & .5 & 3 \\ 0 & -.5 & 5 \end{bmatrix} \rightarrow \begin{bmatrix} 2 & 1 & 2 \\ 0 & .5 & 3 \\ 0 & 0 & 8 \end{bmatrix}$$

where

$$\begin{bmatrix} 2 & 1 & -2 \\ 0 & .5 & 3 \\ 0 & 0 & 8 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} = \begin{bmatrix} 7 \\ -2.5 \\ -8 \end{bmatrix} \text{ again yields } X = \begin{bmatrix} 2 \\ 1 \\ -1 \end{bmatrix} \text{ or } X_1 = 2,\ X_2 = 1,\ X_3 = -1$$

Note that .5 and -.5 are of the same absolute magnitude in the pivot column at the second stage so no switching was necessary. (The operations on the column vector of inhomogeneous terms are not shown as a challenge to the reader.) This is the basic numerical algorithm of linear algebra--forward elimination with partial pivoting.

Elimination with Complete Pivoting

To effect a complete pivotal strategy involves examining all the remaining coefficients for the largest at each pivotal stage and exchanging rows and columns to move that element to the pivotal position. In our system the largest coefficient is 4, the coefficient of X_3 in equation two. Exchanging equation two with equation one and column three with column one we obtain:

$$\begin{aligned} 4X_3 + 0 + X_1 &= -2 \\ 2X_3 + X_2 + X_1 &= 1 \\ -2X_3 + X_2 + 2X_1 &= 7 \end{aligned}$$

This achieves the desired result of getting the largest coefficient in the first pivotal position, but we have had to abandon our matrix notation, as switching columns would have lost the identities of the dependent variables. As before, a) dividing 4 into 2 and multiplying the first equation by .5 and subtracting the first equation from the second; and b) dividing 4 into -2 and multiplying the first equation by -.5 and subtracting the first equation from the third we obtain:

$$4X_3 + 0 + X_1 = -2$$
$$X_2 + .5X_1 = 2$$
$$X_2 + 2.5X_1 = 6$$

Continuing our complete pivotal strategy we examine equations two and three and find we must switch them and also switch columns two and three to place 2.5, the largest coefficient in the second pivotal position obtaining:

$$4X_3 + X_1 + 0 = -2$$
$$2.5X_1 + X_2 = 6$$
$$.5X_1 + X_2 = 2$$

Dividing 2.5 into .5 and multiplying equation two by .2 and subtracting the result from equation three yields

$$4X_3 + X_1 + 0 = -2$$
$$2.5X_1 + X_2 = 6$$
$$.8X_2 = .8$$

By back substitution we again obtain $X_2=1$, $X_1=2$, and $X_3=-1$.

Before we justify this somewhat tiresome detailed explanation, note that we went about our elimination by dividing the pivot into the element to be eliminated, multiplying the pivot row by that quotient and subtracting the result from the row we were operating on. This is important. We are seeking the most efficient algorithm possible--a numerical method that is the best of all possible methods, taking the least amount of steps and producing the most accurate results, given the conditions of the problem. Notice we are not calculating A^{-1} in finding X. We hardly ever need A^{-1}. What we want is $A^{-1}b$. Therefore, in our algorithm we are going to factor A into two triangular matrices L and U such that LU=A. L will be lower triangular with 1's on the diagonal, zeros above. The elements below the diagonal will be precisely those quotients obtained by dividing the element in that position by the pivot element in that column. U will be the upper triangular matrix we obtained prior to starting the back-substitution with the pivot elements on the diagonal, and zeros below. This approach not only allows us to solve any system of n equations in n unknowns in $n^3/3$ operations, but the solution X' for any new right hand side b' can be found in only n^2 operations[2], allowing us to afford extensive iterative gambits with the vector b or the coefficients A. The LU decomposition of A in our first elimination example is:

$$\mathbf{LU} = \mathbf{A} = \underset{\mathbf{L}}{\begin{bmatrix}1&0&0\\1&1&0\\2&1&1\end{bmatrix}} \underset{\mathbf{U}}{\begin{bmatrix}1&1&2\\0&-1&2\\0&0&-8\end{bmatrix}} = \underset{\mathbf{A}}{\begin{bmatrix}1&1&2\\1&0&4\\2&1&-2\end{bmatrix}}$$

If we are given a new right hand side b′, say $\begin{bmatrix}8\\11\\3\end{bmatrix}$ then we simply solve for LC′ = b′

$$\begin{bmatrix}1&0&0\\1&1&0\\2&1&1\end{bmatrix}\begin{bmatrix}C'_1\\C'_2\\C'_3\end{bmatrix} = \begin{bmatrix}8\\11\\3\end{bmatrix} \quad \text{or} \quad C' = \begin{bmatrix}8\\3\\-16\end{bmatrix}$$

and UX′ = C′

$$\begin{bmatrix}1&1&2\\0&-1&2\\0&0&-8\end{bmatrix}\begin{bmatrix}X'^1\\X'^2\\X'^3\end{bmatrix} = \begin{bmatrix}8\\3\\-16\end{bmatrix} \quad \text{or } X' = \begin{bmatrix}3\\1\\2\end{bmatrix} \quad \text{the new solution}$$

The argument for the extremely conservative complete pivotal strategy is somewhat tenuous; one cannot assert, as in the case of Gaussian elimination, that it is the most efficient of all general strategies for a high-speed, digital machine. The most obvious (and trivial) benefit is that it automatically avoids having to deal with a zero pivot. One always has a value in the pivotal position if any coefficients greater than zero remain. This has to be dealt with in any event and complete pivoting does it de rigueur. The product of the diagonal elements of U is the determinant of the matrix A. If a pivot is extremely small (close to 0), the determinant is close to 0, possibly indicating the system is ill conditioned, has a singularity, or is unstable. These problems can be dealt with individually or collectively by inspection, but for an automatic algorithm for all seasons, choosing the largest pivots gives one the most reasonable chance for an accurate, stable solution. The goal is to try to keep the pivots as near the same magnitude as possible. We can do no better than quote Professor Wilkinson, the undisputed authority on the subject: "We do not claim that selecting the largest elements as pivots is the best strategy and indeed this will often be far from true. We do claim that no alternative practical procedure has yet been proposed...complete pivoting, on the other hand, is never a poor strategy."[3] A final comment on this admittedly sketchy argument for complete pivoting has to do with floating

point computations. When numbers are represented in machines by a mantissa m multiplied by 10 raised to some power x ($n=m10^x$ where $.1<|m|<1$), obviously, adding or subtracting two numbers whose exponents differ say, by a factor of two, causes round-off error (.246 + .00193 => .247) if the number of digits of the numbers m are of the order of the word length of the machine. As there are one third of a million operations in the solution of a 100 x 100 matrix ($n^3/3$ steps), round-off error can become a non-trivial factor in the solution if extremely small pivots arise in the course of the computation. It is, therefore, wise to minimize the number of operations and maximize the size of the pivot elements if using floating point numerical operations.

We return now to the solution of the linear system (2) representing the regression model of our surgical tent problem which is accomplished by GAUSS, the program shown in figure 1. First, notice the services afforded the analyst by FM. We DO ^ZIS to allow us to send the result of our efforts to any addressable device, we set DUZ(0)="@" to meet security constraints on our files, and finally we DO ^DICRW to have FM fetch the number of equations (NQ) and the file number (FN) when we respond with 4003 to the "OUTPUT FROM WHAT FILE?" query. The number of variables (NV) are then ascertained by addressing the data dictionary by $P(^DD(FN, 0),"^",4); we subtract 2 from this number as we do not count the name of the equation or the inhomogeneous terms as variables (see equation (4)). Admittedly, one must know what information exists in the tables represented by the ^DD and ^DIC globals but this is exactly analogous to knowing what information we put in our tables in ^DIZ, the user-created files. (Incidentally, all variables beginning with T in GAUSS are temporary such as TXP; the rest of the variables are, hopefully, self-explanatory.)

We create our working version of Ax=b by loading X(J,K) with the coefficient values and the inhomogeneous terms from ^DIZ(FN,J,0). X(J,K) is our relation NORMEQ (File 4003) with domains K, range J.

At this point, we give each instance in the relation a unique name to enable us to switch columns without losing original domain identity--we give each attribute occurrence X(J,K) the name NV*(J-1)+K which translates to an integer which we place in XP(J,K). Consider a 3x3 matrix X with the values X(J,K); if we placed these elements in a string--as the computer does--we have, with the elements numbered:

```
GAUSS     ;DRH/NHRC ; 12/1/87 ; COMPLETE PIVOTAL GAUSSIAN FORWARD ELIMINATION
          D ^%ZIS S DUZ(0)="@",U="^" D ^DICRW S NQ=%,FN=$P(Y,"^")
          S NV=$P(^DD(FN,0),"^",4)-2 I $D(^DD(FN,.001,0)) S NV=NV-1
          F J=1:1:NQ F K=1:1:NV S (LU(J,K),U(J,K),L(J,K))=0
          F J=1:1:NQ F K=1:1:NV+1 S X(J,K)=$P(^DIZ(FN,J,0),"^",K+1),XP(J,K)=
            NV*(J-1)+K
MAIN      F PIV=1:1:NQ-1 D ABSMAX
          F J=1:1:NQ F K=1:1:NV S U(J,K)=X(J,K)
          F J=1:1:NQ F K=1:1:NV I J=K S L(J,K)=1
          G RESULTS
ABSMAX    S XMAX=$$MAX(PIV,NQ,NV,.X)
COLSW     D CSW(PIV,NQ,.X)
ROWSW     D RSW(PIV,NV,.X,.L)
PIVOTS    F J=PIV+1:1:NQ S L(J,PIV)=X(J,PIV)/X(PIV,PIV)
          F J=PIV+1:1:NQ F K=1:1:NV+1 S X(J,K)=X(J,K)-(L(J,PIV)*X(PIV,K))
          Q
RESULTS   F K=NV:-1:1 S ANS(K)=X(K,NV+1)/X(K,K)
          F K=NV-1:-1:1 F J=NV:-1:2 S ANS(K)=ANS(K)-(ANS(J)*X(K,J)/X(K,K))
            Q:J-K=1
          F K=NV:-1:1 D HASH(XP(K,K),NV,.ROW,.COL) S NAME(K)=$P(^DD(FN,COL,0),
            "^",1)
          F K=1:1:NV W !!,"VARIABLE: ",NAME(K),"=",$J(ANS(K),9,6)
L1        D MM(.L,.U,.LU) F J=1:1:NQ W ! F K=1:1:NV W $J(LU(J,K),8,0)
          QUIT
          ;SUBROUTINES: ROWSWITCH,COLSWITCH,MAXELEMENT,MATRIXMULT,HASH
RSW(P,K,X,L) NEW T1,TXP,TRX
          F T1=1:1:K+1 S TXP(P,T1)=XP(P,T1),TRX(P,T1)=X(P,T1),X(P,T1)=
            X(TJ,T1),XP(P,T1)=XP(TJ,T1),X(TJ,T1)=TRX(P,T1),XP(TJ,T1)=TXP(P,T1)
          F T1=1:1:NQ S TRL(P,T1)=L(P,T1),L(P,T1)=L(TJ,T1),L(TJ,T1)=TRL(P,T1)
          Q
CSW(P,J,X) NEW T1,TXP,TCX
          F T1=1:1:J S TXP(T1,P)=XP(T1,P),TCX(T1,P)=X(T1,P),X(T1,P)=X(T1,TK),
            XP(T1,P)=XP(T1,TK),X(T1,TK)=TCX(T1,P),XP(T1,TK)=TXP(T1,P)
          Q
MAX(P,J,K,X) NEW MX,T1,T2,TX
          S MX=-9999999999
          F T1=P:1:J F T2=P:1:K S TX(T1,T2)=X(T1,T2) I TX(T1,T2)<0 S TX(T1,T2)=
            -TX(T1,T2)
          F T1=P:1:J F T2=P:1:K S:TX(T1,T2)>MX MX=TX(T1,T2),TJ=T1,TK=T2
          QUIT K*(TJ-1)+TK
MM(X,Y,Z) NEW M,N,P,R,I,J,K,V,W
          F J=1:1 Q:'$D(X(J,1))  F K=0:1 S V=$O(X(J,K)) Q:V=""
          S M=J-1,N=K
          F J=1:1 Q:'$D(Y(J,1))  F K=0:1 S W=$O(Y(J,K)) Q:W=""
          S R=J-1,P=K I N'=R W !,"# COLS OF A MUST EQUAL # OF ROWS OF B" Q
          F I=1:1:M F J=1:1:P F K=1:1:N S Z(I,J)=Z(I,J)+(X(I,K)*Y(K,J))
          Q
HASH(POS,NV,R,C) I POS<NV!(POS=NV) S R=1,C=POS Q
          I POS#NV=0 S R=POS/NV,C=NV Q
          E  S C=POS#NV,R=(POS-C)/NV+1 Q
```

Figure 1. Gaussian Elimination Algorithm

1	2	3	4	5	6	7	8	9	
X(1,1),	X(1,2),	X1,3),	X(2,1),	X(2,2),	X(2,3),	X(3,1),	X(3,2),	X(3,3)	(6)
Row 1			Row 2			Row 3			

NV (the number of columns of the matrix) times J (the row of the element we are naming) gives us the *maximum* value the *name* can take--if we subtract NV, the number of elements in the row, we obtain the name of the last element in the previous row--now adding K gives us the name (or position if you prefer) that uniquely identifies X(J,K). The linear position of X(3,2), for example, is 3 times 3 (its maximum value possible) minus the number of elements in the row (3) yielding the position of the last element in the previous row, plus 2 (its position in the row under consideration) to yield 8. To undo this is slightly more complex. The reader is directed to the subroutine HASH where the original column is ascertained by manipulating the number of variables (NV) and POS(ition) (value in XP(J,K)).

We now start by finding the largest X(J,K) (in an absolute sense) by defining the extrinsic variable $$MAX and calling subroutine MAX which returns the position name and places TJ and TK in the symbol table (these are the temporary J,K of the largest element of the matrix on the first pivotal pass). The goal being to place this element in X(PIV,PIV), we now switch columns, passing the entire X(J,K) matrix to the subroutine CSW by the call by reference .X . As each column element is manipulated, its name XP(J,K) gets identical treatment. The instruction lines in both CSW and RSW are simple but confusing due to the intermediary temporary positions (TCX, TRX, and TXP) necessary to effect the bookkeeping. Notice in RSW that the elements in our lower triangular matrix L are also shifted to maintain their original functional integrity (i.e. LU=A).

Once we have MAX in the first pivotal position, we do the forward elimination steps in MAIN+7 and MAIN+8. The values of L are computed, and the current matrix X(J,K) has the proper elements eliminated. We then go back to MAIN, increment PIV by 1 and repeat the process until we have U, the upper triangular matrix we need for back-substitution.

The back-substitution begins at RESULTS where we solve for the first unknown--dividing the inhomogeneous term X(K,NV+1) by variable X(K,K): in our present case ANS(3) = X(3,4)/X(3,3). It should be clear at this point we have no idea which variable it is, but happily we have its original column identity in XP(3,3). We go on to divide each of the inhomogenous terms by the approp-

riate pivot (the diagonal elements of U) so that in line RESULTS+1 we can solve for the rest of the variables by back-substitution.

We now call our subroutine (appropriately named) HASH giving it the original name of the column of X(K,K) which is in XP(K,K) and the number of variables (NV) and ask it by reference (.ROW, .COL) to place in ROW, COL, the J,K of X(K,K)'s original position in the matrix A of coefficients. We need only COL, of course, but the identity of the original record could be of interest in other matters. Finally, we go to friendly FM and ask the data dictionary to give us the name of the variable in COL by SET NAME(K)=$P(^DD(FN,COL,0), "^",1) and print the variable names and values with line RESULTS+3. Line L1 of Gauss multiplies the matrices L and U to reproduce the original matrix A as a check of the solution. The subroutine MM first checks to see if the number of columns of the first matrix are equal to the number of rows of the second matrix before attempting the multiplication. This allows the rather elegant matrix array passing notation MM(.L, .U, .LU), otherwise one would have to pass the number of columns of L and the number of rows of U, MM(.L,.U,.LU,C, R), which looks more formidable and does not have the face validity of the more succinct notation. The more responsibility the machine has in these matters, the more accurate the outcome. The line of code that accomplishes the matrix multiplication

```
F I=1:1:M F J=1:1:P F K=1:1:N S Z(I,J)=Z(I,J)+(X(I,K)*Y(K,J))
```

is a reasonable example of MUMPS programming parsimony exploiting the power of the conceptual model, the linearity of the algorithm and the conciseness of the notation.

The solution proceeded as follows, with the original position of the elements shown:

First Pivot:

Original			Column Switch			Row Switch			Elimination		
83,1	25,2	16,3	25,2	8,1	16,3	87,5	25,4	55,6	87,5	25,4	55,6
25,4	87,5	55,6	87,5	25,4	55,6	25,2	8,1	16,3	0	.82,1	.20,3
16,7	55,8	36,9	55,8	16,7	36,9	55,8	16,7	36,9	0	.20,7	1.23,9

Second Pivot:

Column Switch			Row Switch			Elimination		
87,5	55,6	25,4	87,5	55,6	25,4	87,5	55,6	25,4
0	.20,3	.82,1	0	1.23,9	.20,7	0	1.23,9	.20,7
0	1.23,9	.20,7	0	.20,3	.82,1	0	0	.79,1

Back-substituting yields

X1=20191.67, X2=4133.33, X3=758.33

$$L = \begin{bmatrix} 1 & 0 & 0 \\ .63218 & 1 & 0 \\ .28736 & .15888 & 1 \end{bmatrix} \qquad U = \begin{bmatrix} 87 & 55 & 25 \\ 0 & 1.23 & .1954 \\ 0 & 0 & .78505 \end{bmatrix}$$

$$A = LU = \begin{bmatrix} 87 & 55 & 25 \\ 55 & 36 & 16 \\ 25 & 16 & 8 \end{bmatrix} = \begin{bmatrix} 8 & 25 & 16 \\ 25 & 87 & 55 \\ 16 & 55 & 36 \end{bmatrix}$$

and

$$Y' = 20192.67 + 4133.33\ V1 + 758.33\ V2$$

which says that our best estimate is that each operating table adds $4133 to the cost of the surgical tent and each extra light adds $758.

Discussion

The basic ingredients for problem solving with a computer are the computer, the instruction set to run the computer, and some data. Successfully entering information represented by data into a computer brings the analyst face to face with three fundamental issues: (a) where are the data and how do we get there; (b) what are the data; and (c) what do we do with it (assuming we got there! "There is nothing more embarrassing to a database administrator than being asked if his database contains specific information and his replying after a week's examination of the database that he does not know."[4])

The computer machinations are themselves linear--rows of 1s and 0s at one conceptual level. We instruct the computer to sort these finite sequences and we eventually compare the final arrangement to some external criteria to judge the value of the computed information. The models with which we choose to represent reality and those with which we are most comfortable, are in the main, linear. If the data are discrete, as most of the data in our databases are, we have all the power of linear algebra at our disposal. If we choose to display our data in two dimensional "flat" (no pointers--each cell has a discrete value) tables, we have the power of set theory to create unions, intersections, differences, projections, selections, divisions, and joins of our data.

What we strive for in storing data is a consistent, non-redundant structure that permits access to every datum. If the structure is independent of applications used to derive the information from the data, great economy is realized in creating applications, maintaining the database, and in accessing the information by others. Of the techniques (hierarchical/networking and relational) currently in fashion to store data, a relational database management system has the most promise because of its simple, but powerful, theoretical underpinning. It is said that all mathematics can be embedded in set theory. This is a strong inducement to use relational technology. Codd's rule "all information in a relational database is represented explicitly at the logical level in exactly one way--by values in tables"[4], when adhered to, offers the analyst a rational working environment. This rule demands that "all information"--table and attribute information, data integrity constraints, and security information, for example, are available by simply addressing the database with the proper table name, row, and column.

The Veteran's Administration File Manager falls short of this stringent criterium, of course--as do all current relational systems--but FM has incredible strengths. FM "is used to define, enter, and retrieve information from a set of computer-stored files, each of which is described by a 'data dictionary'".[5] In FM a file is a relation, a record is a set, and a field is in a domain.

One creates and defines a file by using option 4--"Modify File Attributes" in the menu driven interface. The file attributes are stored in a table (MUMPS global) named ^DIC. The field attributes defined are stored in a table named ^DD (the "data dictionary"). The data are entered using option 1--"Enter or Edit File Entries" and stored in a table named ^DIZ. The question "where are the data and how do we get there?" is answered by responding with a "?" to the prompt "OUTPUT FROM WHAT FILE?" in the "Print File Entries" option--FM lists all the relations, giving file number, file name, and number of records. Knowing the file number you can ascertain what is there by using option 8 "List File Attributes" and having FM print the data dictionary information for each field in the file. You can get to any stored datum now since you know the file and the field. You simply SET DATUM=$P(^DIZ(FILENUMBER, RECORDNUMBER,0),"^",FIELD NUMBER).

The question "what are the data?" was answered when you created the file. Each field is allowed one of seven data types--date, numeric, a set, free

text, word processing, computed ("virtual" field), and a pointer. In addition to the data type, range constraints and transaction restrictions can be set to further ensure data integrity."What do we do with it?" is a function, of course, of what "it" is. Multiple regression has to do with numbers, so we do arithmetic. We exploit the linearity of the machine, of the database model, and the notation by generalizing the problem. We let X stand for the tables, J for the records, and K for the fields. We can now manipulate X(J,K) as a relation (set theory) or as an attribute value (linear algebra).

The case for FM as a relational database management system is very strong. There is no doubt in the author's mind that the relational "view" is going to dominate the future in data structuring. FM has all the utilities necessary to do the job, and it is just a matter of time before someone writes a structured query language for this monumental piece of public domain software to enhance an already incredibly useful tool.

FM is written in MUMPS, which brings us back to our original question: Is MUMPS a reasonable choice for numerical analysis in statistical work?

One of the bugbears in statistical analysis is the problem of missing data leading to underestimation (of the mean, for example, when N is taken as the number of sets and ΣX does not include values for some members). Conventionally, the delimiter in MUMPS globals is the circumflex ("^"), defining what exists between ^s as an attribute value in a relational database. "No data" can take the form of null (^^), the empty string (^""^), or the empty set (^0^). The universe of possible attribute representations is a combination of one or more of the 128 definitions (represented by the integers from 0 to 127) in the United States American Standard Code for Information Interchange (ASCII) character set.

Consider this line of MUMPS code:

```
F J=1:1:NV S N(J)=NR F K=1:1:NR I X(J,K)']$C(0) S N(J)=N(J)-1
```

where

J = the number of the domain (field number or variable, however you are picturing your data)

NV = the last domain to be examined (the number of variables)

K = the set number (record)

NR = the range of J (the number of records)

I = IF

X = the relation (the table name or file name)

' = not

] = follows, in the sense of the collating order of the ASCII character set

$ = signals what follows is a special instruction--evaluate the parenthetical argument

C = the symbol for the function that takes the integer value argument and returns the ASCII definition it represents (some of which are not printable).

0 = the decimal ASCII value for NULL, the first in the code which ranges from 0 to 127

S = "Set" in MUMPS

N = the number of sets to be examined.

This is an existence theorem. It states "set N for the field you are examining equal to the number of records you are going to examine. From the first record to the last, look at each field. If there is no value for that record and field, reduce N by one."

I X(J,K)']$C(0) says, if the contents of X(J,K) does not follow the first value in the ASCII code (which is null), then the contents must be null--if it ain't in the universe, it doesn't exist! (Remember, by definition, "" and 0 do exist, of course, and one can evaluate them in whatever context they occur. The empty string might mean the subject refused comment; zero could be his/her response to number of children.) Typically the time to use this powerful screen is when one is loading the working data subset into resident memory.

At this time one should also transpose the data (turn the table on its side--$X^T(J,K) \Rightarrow X(K,J)$) so that the rows consist of the variables and the columns are the records in which the variables occur. The discussion of the existence rule followed the transposed concept. This is also consistent with mathematical notation for a function which is a relation on coupled pairs:

$$f(x) = R(x,f(x)) = R(x,y) = y$$

where x is the domain, y is the range, R is the relation, and f signifies a function or rule on x ($f \equiv R$). It takes three times as long to sum the elements of a matrix (X(J,K)) by

```
F K=1:1:NCols F J=1:1:NRows S S=S+XT(J,K)
```

than it takes doing

```
F K=1:1:NCols F J=1:1:NRows S S=S+XT(K,J)
```

on a VAX 11/785. Essentially, what the second line amounts to is reading all the descendants of a node before preceding to the next node. If one does not

transpose the data, the machine has to repeatedly skip from node to node for each datum (see the linear array in equation (6)). An analogy is the process of shopping in a supermarket. You go down the aisles (records) and pick from the bins (fields). You seldom shop aisle 1 bin 1, then aisle 2 bin 1 to aisle 12 bin 1 and then start over at aisle 1 bin 2, etc. Unfortunately, poor programming strategies like this occur all too often in the misuse of MUMPS and lead to opinions MUMPS is slow and unsuitable for scientific work. The penalties are severe in local memory but if extended to reading globals the increase in disk reading time is astronomical. (The ratio of the two approaches remains 3:1, but the time goes from milliseconds to minutes.)

The power and conciseness of the MUMPS language is without parallel in the routine for deriving the square root of positive real numbers:

```
      S Y=0 Q:X'>0  S Y=1+X/2
L     S T=Y,Y=X/T+T/2 G L:Y<T
```

and in this Shell sort (ascending order) routine[1]:

```
     S G=N\2
S2   F I=G+1:1:N S L=I-G I (Y(J,L)>Y(J,L+G)) S T=Y(J,L),Y(J,L)=Y(J,
     L+G),Y(J,L+G)=T G S2
     S G=G\2 Q:G'>0 G S2
```

which divides a list Y of N items into two halves, compares, in order, the elements of the first half with the elements of the second half, and switches them if the first is greater than the second. After the first ordering the list is divided into quarters and the process repeated; finally adjacent elements are compared completing the ordering.

These examples strongly support the contention MUMPS can be used procedurally in a numerical context. If the analyst clearly differentiates between storing things and manipulating them, efficient algorithms can be written in MUMPS. File structuring, sensible blocking on disks, emphasis on exploiting the size of the partition, and structured programming will do a lot to dispell the notion MUMPS is slow when manipulating numbers.

The emphasis on networking in the sixties stemmed from the electrical engineers who were our first programmers. This influence is seen in the idea of embedding data in the subscript of a MUMPS global. The path dependencies resulting have proven to be almost intractable for analysts. The current progress in compiling MUMPS[6] lends encouragement to those writing code that is linear in nature, thus producing efficient object code.

The fact that ISM utilizes 19 fixed point decimal digits on DEC machines is a numerical analyst's dream--this is automatic double precision--when compared to the round-off error nightmares the fifties and early sixties with eight and ten digit floating point machines. String manipulation features like $F and $L allow incredible shortcuts to examining numbers, cutting the code necessary to write a logarithm routine, for instance, to a dozen lines--that are as fast as all but the most strongly typed languages like PASCAL--certainly comparible to FORTRAN if the I/O overhead is considered. (This is intuitive; benchmarking comparisons would be interesting, using compiled MUMPS.)

The future of MUMPS--its longevity--is assured. The marriage of PROLOG with MUMPS in GNOSIS[7] is based on MUMPS string manipulation capabilities, global data structure, and indirection facility. The use of MUMPS in database management is a function of its subscripting and efficient global structuring. The parameter passing facilities now available make possible maximally compact code. MUMPS is a language of the future if its strengths are exploited and not denigrated by unintelligent use.

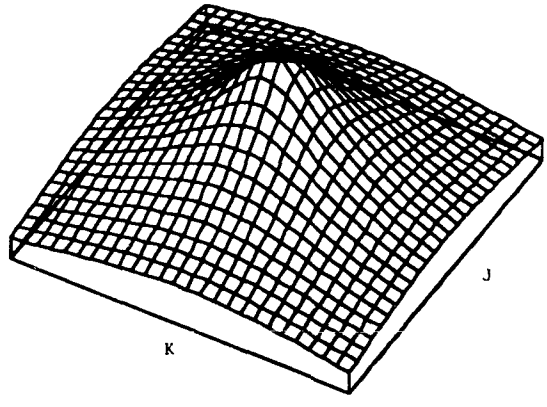


Figure 2. Bi-Variate Normalized Gaussian Distribution

Conclusion

The simplicity of the relational data structure can lead to powerful analytical processes. If one can visualize figure 2 as a K-J plane with the value of the X(K,J) attribute represented by the height of the curve above the

point K,J, one has a three-dimensional view of the data distribution. Figure 2 happens to be the bi-variate normal Gaussian distribution, but distribution functions can be fitted to real data in columns of relational tables and easily graphed yielding dramatic insight into the data relations and distributions over the range of the domains.

Two roles for the symbols K and J are being utilized: (1) K and J as a coupled pair (of integers, for example) to locate a value of a point in a plane; (2) K and J as the place holders for the name of a column in a table and the name of a row in that table.

If we designate the table as X, we can write X(K,J) to describe the table X with domain K, range J. This now allows us to treat the X in set theoretic fashion employing all the operations of that branch of mathematics--it does not matter what the K is at this level as long as some K in the set we are operating on are like some K in other sets (for certain operations).

If we assign cardinal numbers to K and J, we can manipulate the values that each pair represent algebraically, as we did in GAUSS, arriving at numbers called statistics which purport to say something about how the attributes in a domain are related to some criteria--say the mean of all the salaries, or predict the contribution of different elements to the price of a surgical tent as a linear combination of the domains involved when those domains are combined to form a basis for a vector space (least squares solution to an overdetermined system of equations).

The relational database structure is sound mathematically. It allows us to gracefully handle issues of data integrity, independence, and access. When viewed as presented here, it can be exploited with all the power of set theory and linear algebra, which in turn leads to fast, efficient machine code.

Theoretical Implications

There are at least three emerging areas that the use of MUMPS, relational technology, and the underlying mathematical calculus, are eminently suited. The first, in conceptual simplicity, is distributed parallel processing. It is not difficult, analytically or from the hardware standpoint, to visualize two or more 80386 processors operating on the columns of matrices. One could compute every conceivable statistic, probability density function, or sort almost instantaneously on each column.

The second theoretical consideration is the use of MUMPS and set theory in knowledge based systems--so called artificial intelligence. The indirection feature of MUMPS that allows the program to write code *modifying* itself is of enormous import in problems involving servo mechanisms, cybernetics, analysis, and decision making (there is no free lunch; this same facility exacts a price in the compilation of the code). The string manipulating features of MUMPS easily duplicate LISP features with appropriate coding. The local/global storage structure of MUMPS reduces I/O to child's play and is maximally efficient in arrangement and space. The choice of MUMPS by the Japanese to drive PROLOG in GNOSIS in creating a fifth generation language was not whimsical.

The third body of ideas beginning to create widespread interest is the paradigm of computational reflection[8]. This concept is illustrated in this paper in the programming to keep track of the original domain of the elements of the matrix A. Another example is

```
F J=1:1 Q:'$D(X(J,1))  F K=0:1 S V=$O(X(J,K)) Q:V=""  S M=J-1,N=K
```

in the subroutine MM where we ask the machine to determine if X(J,K) is a proper object (i.e. checking to see if the number of columns of matrix A equal to the number of rows of matrix B) for the operation to follow and to inform us if it is not. This might be thought of as meta-programming. It is a program about the program. These examples illustrate exactly what an object oriented programming schema would accomplish--we have linked meta-information about the object to the manipulation of the object. To be able to write commands based on what has happened in the meta phenomena, as we can in MUMPS by using indirection, allows the programmer an unprecedented level of control.

Conceptually, MUMPS globals are inherently relational. The Veteran's Administration File Manager is essentially a collection of information in MUMPS globals. Relational database structures are two dimensional tables that are most efficiently manipulated by set theory. The entities of the tables are amenable to the rigors of linear algebra. The resolution of a classical problem in numerical linear algebra demonstrates the ease, simplicity, and efficiency of the use of MUMPS, FM, and relational techniques.

The choice of a computational language that contains the tools needed for innovative research in a mathematically sound manner is prudent. MUMPS, FM, and relational concepts offer that virtue.

Author's Note: Professor Strang's delightful book "Linear Algebra and Its Applications" was relied upon heavily in this paper. The numerical example for Gaussian elimination is from that book and was used for ease of verification. Professor Wilkinson's two classics "The Algebraic Eigenvalue Problem" and "Rounding Errors in Algebraic Processes" served as authority to support the remarks on the process and to provide examples to check the error bounds on the algorithm. The author lays modest claim to the hashing stratagem as an original contribution.

REFERENCES

1. Hodgins, D.R. Descriptive Statistics Using the Veterans' Administration File Manager as a Relational Database Management System, NHRC Report No. 87-22, 1987.

2. Strang, G. Linear Algebra and Its Applications. Academic Press, Inc., New York, NY, 1976.

3. Wilkinson, J.H. The Algebraic Eigenvalue Problem. Oxford University Press, Amen House, London E.C. p. 217, 1965.

4. Codd, E.F. Is Your DBMS Really Relational?, ComputerWorld, Part 1, October 14, 1985.

5. Timson, G. VA Fileman Version 17 Release Notes, 1986, personal communication.

6. Dayhoff, R. New Developments in MUMPS. MUMPS News, Vol 5:1, Feb 1988.

7. Specification of Language GNOSIS. MUMPS System Laboratory, 39-15 Daikan-cho, Higashi-ku, Nagoya, Japan 461, 1986.

8. Maes, P. Concepts and Experiments in Computational Reflection. AI LAB Vrije Universiteit Brussel, Pleinlaan 2, B-1050 Brussels, 1987.

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION Unclassified	1b RESTRICTIVE MARKINGS None
2a SECURITY CLASSIFICATION AUTHORITY N/A	3 DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b DECLASSIFICATION / DOWNGRADING SCHEDULE N/A	
4 PERFORMING ORGANIZATION REPORT NUMBER(S) NHRC Report No. 88-12	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION Naval Health Research Center 6b OFFICE SYMBOL (If applicable) Code 20	7a NAME OF MONITORING ORGANIZATION Commander, Naval Medical Command
6c ADDRESS (City, State, and ZIP Code) P. O. Box 85122 San Diego, CA 92138-9174	7b ADDRESS (City, State, and ZIP Code) Department of the Navy Washington, DC 20372
8a NAME OF FUNDING / SPONSORING ORGANIZATION Naval Medical Research & Development Command 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c ADDRESS (City, State, and ZIP Code) Naval Medical Command National Capitol Region Bethesda, MD 20814-5044	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO: 63706N PROJECT NO: M0095 TASK NO: .005 WORK UNIT ACCESSION NO: 1053

11 TITLE (Include Security Classification)

(U) COMPLETE PIVOTAL GAUSSIAN ELIMINATION USING THE VETERAN'S ADMINISTRATION FILE MANAGER

12 PERSONAL AUTHOR(S)
Dallas R. Hodgins

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Final	FROM _____ TO _____	88 March 16	28

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	
			Gaussian Elimination, VA File Manager, Linear Algebra

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

The choice of a programming language in research and development is a serious commitment of time, money, and personnel. The principal question addressed is whether MUMPS is a viable language for developing statistical models that require numerical analysis. An algorithm to solve the linear equations of multiple regression is developed to explore and exploit the characteristics of the MUMPS language in handling numbers. Also of interest is the selection of a database management system and the conceptual schema of the data structure. Files structured relationally in the Veterans' Administration Filemanager are manipulated algebraically.

The development of concise, efficient language constructs lend support to using MUMPS in small sample analysis in statistical research. At one level, statistical analysis deals with discrete values in a two-dimensional space. Assigning values to the points of this space is the equivalent of a function over the space. If one has a function (coupled pairs), the application of set theory is obvious. To most efficiently use set theory and the power of the notation of linear algebra, the data should be structured relationally.

20 DISTRIBUTION / AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☐ DTIC USERS	Unclassified
22a NAME OF RESPONSIBLE INDIVIDUAL Dallas R. Hodgins	22b TELEPHONE (Include Area Code) 619/553-9291 22c OFFICE SYMBOL Code 20

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted
All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE

☆U.S. Government Printing Office: 1985-607-047

19. (Cont'd) The MUMPS language, in conjunction with the Veteran's Administration File-manager used to create relational data structures, is a powerful, versatile media for numerical analysis. Using the notation of linear algebra and properly exploiting the inherent data storage virtues of MUMPS leads to efficient, fast code as is exemplified in the algorithm for complete pivotal forward Gaussian elimination presented in this paper.

END

DATE

FILMED

9-88

DTIC